School of Mathematical and Computational Sciences

Indian Association for the Cultivation of Science

Compiler Construction: COM 5202

Tutorial V (04 February 2026)

M. Sc Semester IV: 2025-2026 *Instructor:* Goutam Biswas

Exercise 1. **[10]**

Consider the following grammar.

$$
\begin{array}{rcl}
P & \rightarrow & \textbf{prog}\ DLO\ \textbf{start}\ CL \\
DLO & \rightarrow & DLO\ DL \mid \varepsilon \\
DL & \rightarrow & TY\ VL \\
TY & \rightarrow & \textbf{int} \mid \textbf{real} \\
VL & \rightarrow & VL\ \textbf{id} \mid \textbf{id} \\
CL & \rightarrow & CL\ C \mid C \\
C & \rightarrow & IC \mid OC \mid AC \\
IC & \rightarrow & \textbf{in id} \\
OC & \rightarrow & \textbf{out}\ E \mid \textbf{out str} \\
AC & \rightarrow & \textbf{id :}\ E \\
E & \rightarrow & E + T \mid E - T \mid T \\
T & \rightarrow & T * F \mid T / F \mid F \\
F & \rightarrow & \textbf{ic} \mid \textbf{rc} \mid \textbf{id} \mid (\ E\ )
\end{array}
$$

The terminals are { **prog, start, int, real, in, out, ic, rc, id str, :,** +, −, ∗, /, (,) }.
The non-terminals are { *P, DLO. DL, TY, VL, CL, C, IC, OC, AC, E, T, F* }.
The start symbol is P.
The regular expressions are,
id: `[A-Z][\_a-zA-Z0-9]*`
ic: `0|[1-9][0-9]*`
rc: `[0-9]+\.[0-9]*((e|E)[+-]?[1-9][0-9]?)?`
str: `(\"[^\"]*\")`
Comment: `(\/\/.*\n)`

Write flex specification for the set of terminals of the grammar. The name of the flex file should be `lex.l`. *The header file for it is* `y.tab.h`. *The header file defines some of the token values and the type of* `yylType`.
We take the ASCII codes as token values for all single character terminals.

The scanner ignores a comment, but keeps track of the line number.

```
// y.tab.h
#ifndef _Y_TAB_H
#define _Y_TAB_H

#define ID     300
#define IC     301
#define RC     302
#define STR    303
#define PROG   304
#define START  305
```

```
#define REAL   306
#define INT    307
#define IN     308
#define OUT    309
#define NOTOK  400

int yylex(void);
typedef union {
        int integer ;
        double real;
char *string;
} yylType;

#endif
```

The scanner function `yylex()` *will be called from* `main()` *in* `myLex.c` *(include* `y.tab.h` *in* `myLex.c`*). The* `main()` *function calls* `yylex()` *and prints the stream of tokens (shown in Input/Output). You may use the following* `Makefile` *for the process of compilation.*

The input is from the `stdin`*. You may write the input in a file and redirect it:*

```
$ ./a.out < data
```

Makefile:

```
objfiles = myLex.o lex.yy.o

a.out:  $(objfiles)
        cc $(objfiles)

myLex.o:        myLex.c y.tab.h
                cc -c -Wall myLex.c

lex.yy.o:       lex.yy.c
                cc -c lex.yy.c

lex.yy.c:       lex.l y.tab.h
                flex lex.l

clean:
                rm a.out lex.yy.c $(objfiles)
```

Finally prepare a `.tar` *file using the following command.*

```
$ tar cvf <roll no>.5.tar y.tab.h lex.l myLex.c Makefile
```

Send the `.tar` *file to* `goutamamartya@gmail.com`*.*

Input-Output

```
$ ./a.out
prog
<prog>
int
```

```
<int>
Factorial
<ID, Factorial>
factorial
Invalid characters: factorial
1234
<IC, 1234>
1234.567e-2
<RC, 12.345670>
"Indian Association for
the Cultivation of
Science"
<STR, Indian Association for
the Cultivation of
Science>
Value1: A1+B2*50/(X5-11)
<ID, Value1> <:> <ID, A1> <+> <ID, B2> <*> <IC, 50> </> <(> <ID, X5> <-> <IC, 11> <)>
```

Input from a data file.

```
// The file name is d0
prog
  int N
  real X1
start
  out "Give an integer"
  in N
  X1: N/13 - N*12
  out X1
```

Output

```
$ ./a.out < d0
<prog>
<int> <ID, N>
<real> <ID, X1>
<start>
<out> <STR, Give an integer>
<in> <ID, N>
<ID, X1> <:> <ID, N> </> <IC, 13> <-> <ID, N> <*> <IC, 12>
<out> <ID, X1>
```

Exercise 2. *Consider the context-free grammar*

$$S \to S\ S\ +\ |\ S\ S\ *\ |\ a$$

and the string $aa + a*$

(a) *Give a leftmost derivation for the string.*

(b) *Give a rightmost derivation for the string.*

(c) *Give a parse tree for the string.*

(d) *Is the grammar ambiguous or unambiguous?*

(e) *Describe the language generated by the grammar.*

Exercise 3.

(a) *Write the language of the grammar* $S \rightarrow 0\ S\ 1 \mid 0\ 1$ *in set notation.*

(b) *Remove left-recursion for the grammar* $S \rightarrow S\ S\ +\ \mid S\ S\ *\ \mid a$ *and then left-factor it.*

(c) *Show that the grammar* $S \rightarrow (S) \mid SS \mid \varepsilon$ *is ambiguous. Is there any equivalent unambiguous grammar?*

Exercise 4. *We extend the CFG production rules as follows: regular expressions of grammar symbols are used in the right-hand side of production rules with proper meta symbols. Does this extension of notation extends the class of languages beyond CFL?*

Exercise 5. *Consider the following grammar* G_5 *with terminal symbols* $\{+, *.(,), 0, 1\}$ *and* E *as the start symbol.*

$$\begin{aligned} E &\rightarrow E\ +\ T \mid T \\ T &\rightarrow T\ F \mid F \\ F &\rightarrow F\ * \mid P \\ P &\rightarrow (\ E\ ) \mid 0 \mid 1 \end{aligned}$$

(a) *Informally define* $L(G_5)$.

(b) *Remove left-recursion and form the grammar* G_{5b}.

(c) *Find First() and Follow() of different non-terminals.*

(d) *Draw the parse tree corresponding to* $0 * +(01)*$ *following the First and Follow information.*